DISEÑO DE PROCESADORES DEDICADOS

Práctica 6
"LCD de Propósito General"

Dr. Juan Carlos Herrera Lozada
jlozada@ipn.mx

Instituto Politécnico Nacional



Centro de Innovación y Desarrollo Tecnológico en Cómputo
CIDETEC

Mayo 2015

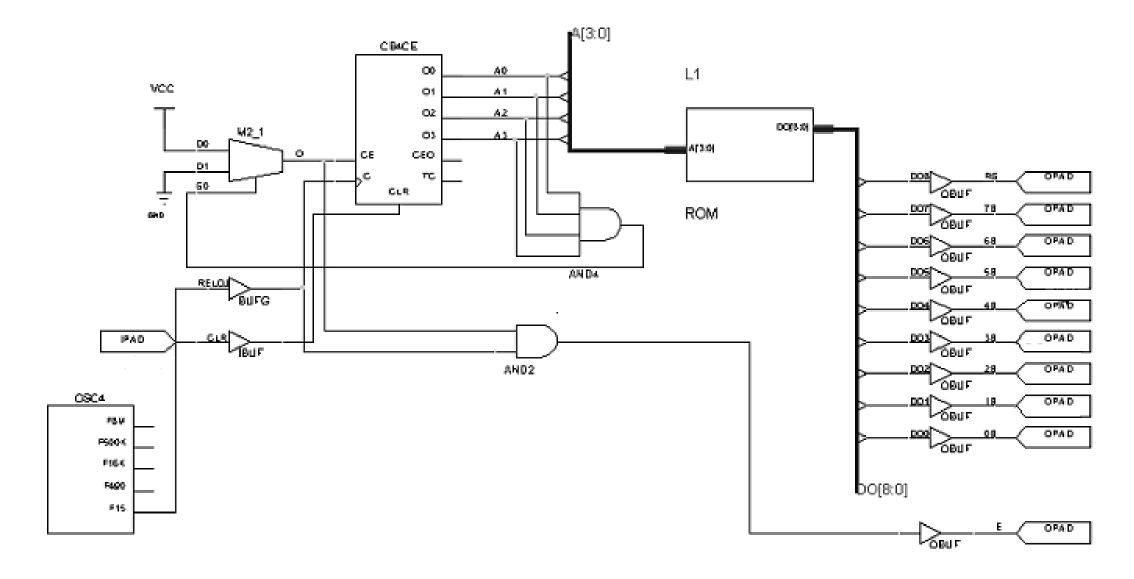
Campo 1: **Datos Personales.**

Campo 2: **Objetivos.**

- Síntesis Lógica y Programación de FPGA.
- Despliegue de datos en una LCD de propósito general.

Campo 3: **Desarrollo de la Práctica.**

1. (3 puntos) Revisa el documento "LCD de Propósito General" en la página de la asignatura. El siguiente diagrama básico (el cual se anexa sólo como referencia) ejemplifica una ROM que inicializa y despliega un mensaje corto en una LCD de entrada paralela (14 pines), similar a la que incluye la tarjeta de desarrollo. En este diagrama, el bloque OSC4 implica un módulo divisor de tiempo (frecuencia).
Observa que el multiplexor permite deshabilitar el contador y la señal del reloj para el *Enable* de la LCD, tras alcanzar el último estado del conteo ("1111"), evitando que se reinicie la escritura del mensaje en la LCD y garantizando que en una sola ocasión se realice el proceso de escritura.



Caso práctico 1: Tarjeta Spartan 3A/3AN 3S700A.

Un código sencillo en VHDL para desplegar el mensaje "CIDETECipn" en una LCD se lista a continuación. Este código está incluido en el proyecto completo para la tarjeta *Spartan 3A / 3AN, 3S700A* de Digilent, que se encuentra disponible en la misma página de la asignatura.

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
use IEEE.std_logic_arith.all;
```

```
use IEEE.std_logic_unsigned.all;

ENTITY ini_lcd IS
  PORT(
         RELOJ, RESET: in STD_LOGIC;
         PASSCLK: out STD_LOGIC;       -- Salida del PIN Enable de la LCD
         lcd: out STD_LOGIC_VECTOR (8 downto 0));
  END ini_lcd;

ARCHITECTURE mensaje OF ini_lcd IS
signal CLK: STD_LOGIC;
signal COUNT: STD_LOGIC_VECTOR (3 downto 0);
begin
contador: process (CLK, RESET)
begin
   if RESET='0' then
      COUNT <= "0000";
   elsif CLK='1' and CLK'event then
         if COUNT /= "1111" then
               COUNT <= COUNT + 1;
         else
               COUNT <= COUNT;
         end if;
    end if;
end process contador;

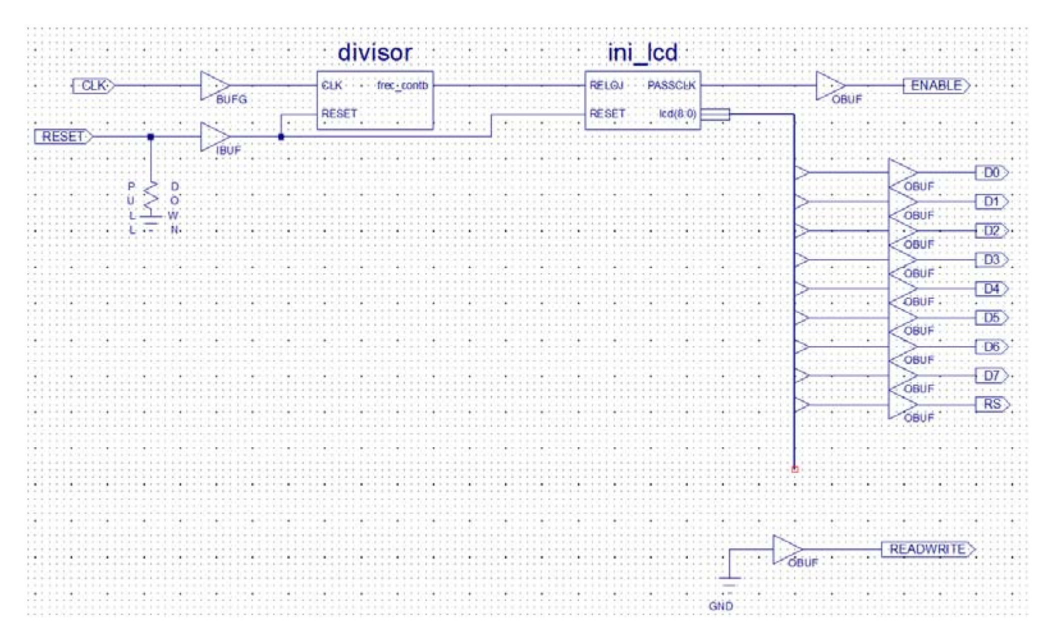
mux: process (COUNT, RELOJ, CLK)
begin
-- Este MULTIPLEXOR permite deshabilitar el RELOJ
if COUNT /= "1111" then
CLK <= RELOJ;
else
CLK <= '1';
end if;
PASSCLK <= CLK;
end process mux;


pantalla: process (COUNT)
begin
--      RS|D7|D6|D5|D4|D3|D2|D1|D0
--      RS es el MSB de la palabra y se conecta al pin 4 de la LCD.
--      D7 es el MSB del dato y se conecta en el pin 14 de la LCD, por lo que el LSB
--      D0 se conecta al pin 7 de la LCD.
case COUNT is
when "0000" => lcd <="000111000"; --0 38
when "0001" => lcd <="000000110"; --0 06
when "0010" => lcd <="000001110"; --0 0E
when "0011" => lcd <="000000001"; --0 01     Clear Screen
when "0100" => lcd <="010000000"; --0 80
when "0101" => lcd <="101000011"; --1 43     "C"
when "0110" => lcd <="101001001"; --1 49     "I"
when "0111" => lcd <="101000100"; --1 44     "D"
when "1000" => lcd <="101000101"; --1 45     "E"
when "1001" => lcd <="101010100"; --1 54     "T "
when "1010" => lcd <="101000101"; --1 45     "E"
when "1011" => lcd <="101000011"; --1 43     "C"
when "1100" => lcd <="101101001"; --1 69     "i"
when "1101" => lcd <="101110000"; --1 70     "p"
when "1110" => lcd <="101101110"; --1 6E     "n"
when others => lcd <="100010000"; --1 10     " "

end case;
end process pantalla;
END mensaje;
```

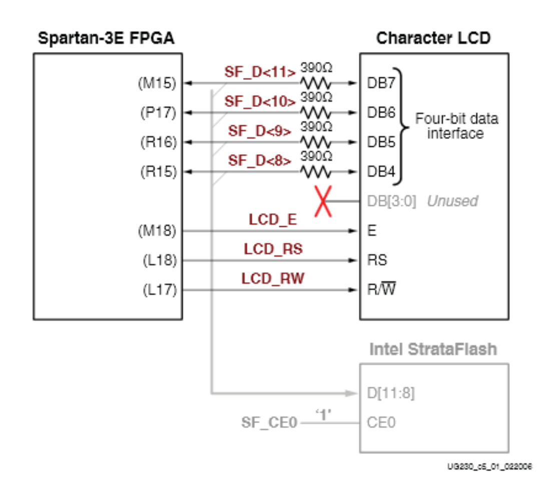
El diagrama esquemático, después de haber generado un símbolo esquemático para el código anterior, se muestra a continuación. Obsérvese que el símbolo esquemático para el divisor de tiempo se debe crear previamente (también está contenido en el proyecto completo que se puede consultar como referencia). Recuerda que para nuestros propósitos prácticos, estamos asignando el pin R/W de la LCD a un nivel lógico bajo (GND) para que se permita la escritura de datos de manera continua, por lo mismo, en el diagrama esquemático se aprecia dicha conexión.

Observa que los primeros cinco estados del contador del código VHDL (cuando *RS* está puesto a '0') que permiten la configuración de la pantalla, no están comentados. Debes revisar la documentación de la LCD (ya sea utilizando el apunte previo o en el mismo manual de usuario de la tarjeta) para saber qué configuración está definiendo cada palabra enviada a la LCD e incluir el comentario pertinente en la respectiva línea del código. Puedes cambiar el texto del mensaje para realizar otra prueba.



Caso práctico 2: Tarjeta Spartan 3S500E.

La LCD incluida en la tarjeta *Spartan 3S500E* trabaja de un modo diferente a la anterior. También se trata de una LCD de 2 líneas con 16 caracteres cada una, pero que utiliza un esquema de sólo 4 bits de datos en vez de los 8 que usa la tarjeta *3S700A*. Observa el siguiente diagrama extraído del manual de usuario de la tarjeta *3S500E*.



Se observa que sólo se utiliza el *nibble* más significativo de los datos (DB4, DB5, DB6, DB7) y el *nibble* menos significativo DB(3:0) no se utiliza.

Antes de explicar cómo se envían los datos con la conexión de 4 bits, es importante que observes que en el diagrama anterior se tiene una señal etiquetada como *SF_CEO* y que está puesta a VCC ('1' lógico) y que forma parte del bloque denominado *Intel StrataFlash* visible en la parte inferior del mismo diagrama; lo anterior es necesario para deshabilitar el acceso compartido de los datos a la memoria *StrataFlash* y así tendremos acceso completo de *Read/Write* sólo a la LCD (revisar la página 42 del manual de usuario de la tarjeta de desarrollo en comento). En su momento, en el archivo de definición del pinout (*.ucf) anotarás *NET "SF_CE0" LOC = "D16"* que es el pin asignado en el FPGA para la señal *SF_CEO.*

Dr. Juan Carlos Herrera Lozada
jlozada@ipn.mx
CIDETEC IPN, México, 2015

Debido a la interfaz de sólo 4 bits, el envío de la palabra de configuración y/o datos se tiene que hacer en dos tiempos. Por ejemplo, si quisiéramos aplicar *Clear Screen*, y recordando el set de instrucciones de configuración para la LCD, debemos enviar a la LCD el dato hexadecimal 01 (00000001), manteniendo RS en '0' lógico al tratarse de un comando de configuración; para la tarjeta *3S500E* se requiere primero enviar 0000 y después 0001, de manera sincronizada y continua para que el control de la LCD reconozca los 8 bits.

El siguiente código escribe el mensaje *CIDETECipn* en la LCD de la tarjeta *3S500E*. Observa que el contador binario es de 5 bits, para tener 32 estados, dado que se requieren dos estados por cada palabra que se envía a la LCD y por ende es un esquema de doble duración en comparación al código utilizado para la tarjeta *3S700A*, no obstante, notarás que los bloques de control (contador, multiplexor y deccoder) son muy similares al ejercicio anterior.

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
ENTITY ini_lcd IS
  PORT(
         RELOJ, RESET: in STD_LOGIC;
         PASSCLK: out STD_LOGIC;       -- Salida del PIN Enable de la LCD
         RS, D7, D6, D5, D4: out STD_LOGIC);
  END ini_lcd;

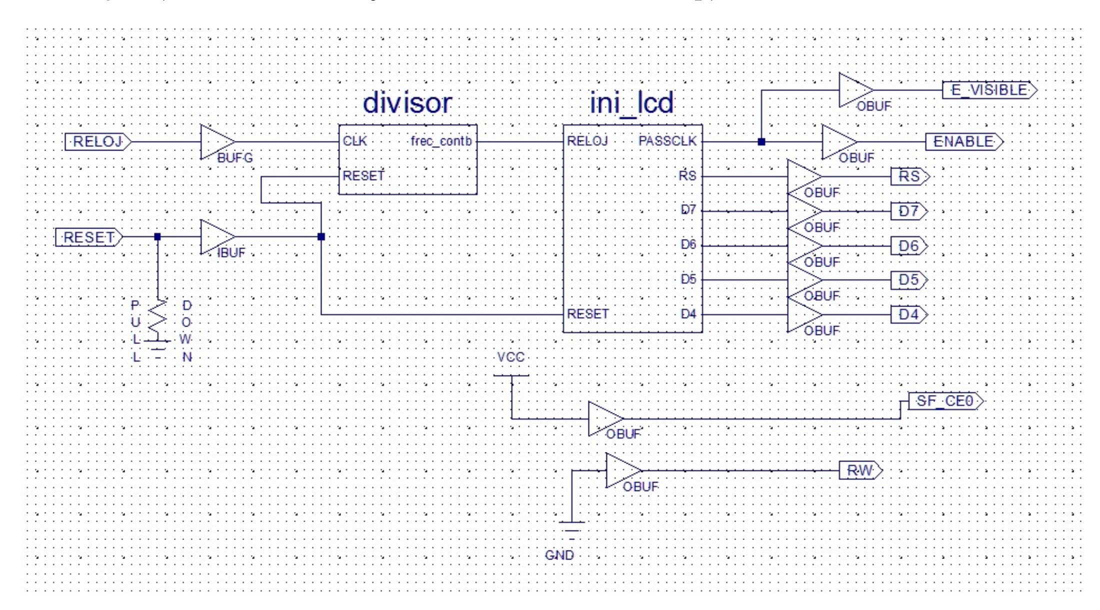
ARCHITECTURE mensaje OF ini_lcd IS
signal CLK: STD_LOGIC;
signal COUNT: STD_LOGIC_VECTOR (4 downto 0);
signal lcd: STD_LOGIC_VECTOR (4 downto 0);
begin
contador: process (CLK, RESET)
begin
   if RESET='0' then
      COUNT <= "00000";
   elsif CLK='1' and CLK'event then
         if COUNT /= "11111" then
               COUNT <= COUNT + 1;
         else
               COUNT <= COUNT;
         end if;
    end if;
end process contador;

mux: process (COUNT, RELOJ, CLK)
begin
-- Este MULTIPLEXOR permite deshabilitar el RELOJ
if COUNT /= "11111" then
CLK <= RELOJ;
else
CLK <= '1';
end if;
PASSCLK <= CLK;
end process mux;

pantalla: process (COUNT)
begin
--      RS|D7|D6|D5|D4|
--      RS es el MSB de la palabra y se conecta al pin 4 de la LCD.
--      D7 es el MSB del dato para la LCD.
case COUNT is
when "00000" => lcd <="00000"; --0 06
when "00001" => lcd <="00110";
when "00010" => lcd <="00000"; --0 0E
when "00011" => lcd <="01110";
when "00100" => lcd <="00000"; --0 01     Clear Screen
when "00101" => lcd <="00001";
when "00110" => lcd <="01000"; --0 80
when "00111" => lcd <="00000";
when "01000" => lcd <="10100"; --1 43     "C"
when "01001" => lcd <="10011";
when "01010" => lcd <="10100"; --1 49     "I"
when "01011" => lcd <="11001";
when "01100" => lcd <="10100"; --1 44     "D"
```

```
when "01101" => lcd <="10100";
when "01110" => lcd <="10100"; --1 45     "E"
when "01111" => lcd <="10101";
when "10000" => lcd <="10101"; --1 54     "T "
when "10001" => lcd <="10100";
when "10010" => lcd <="10100"; --1 45     "E"
when "10011" => lcd <="10101";
when "10100" => lcd <="10100"; --1 43     "C"
when "10101" => lcd <="10011";
when "10110" => lcd <="10110"; --1 69     "i"
when "10111" => lcd <="11001";
when "11000" => lcd <="10111"; --1 70     "p"
when "11001" => lcd <="10000";
when "11010" => lcd <="10110"; --1 6E     "n"
when "11011" => lcd <="11110";
when "11100" => lcd <="01000"; --0 80
when "11101" => lcd <="00000";
when others => NULL;
end case;
RS <= lcd(4);
D7 <= lcd(3);
D6 <= lcd(2);
D5 <= lcd(1);
D4 <= lcd(0);
end process pantalla;
```

El diagrama para este ejercicio quedará como se aprecia a continuación. El archivo *LCD_500E.zip*, en la página web del curso, contiene los códigos en VHDL para los símbolos *ini_lcd* y *divisor*, así como el archivo de definición de pines *(*.ucf)*. En el diagrama *E_VISIBLE* es el *ENABLE* que se conecta a un LED de la tarjeta, mientras que *ENABLE* se dirige al pin respectivo de la LCD. El pin *RW* de la LCD, así como *SF_CEO*, ambos con un número específico en la tarjeta (ver el archivo **.ucf* contenido en *LCD_500E.zip*), se conectan a *GND*.



2. **(7 puntos)** Diseña un contador 00-99, ascendente - descendente. Este contador debe visualizarse en la LCD de la tarjeta de desarrollo.
Requieres incluir tres variables en tu diseño: *Reset*, *Stop* y *Dirección* (para seleccionar si el conteo es ascendente o descendente). Debes inicializar tu pantalla, escribir un mensaje que diga “CUENTA:” y posteriormente desplegar el contador.

Puedes auxiliarte, si así lo consideras necesario, del siguiente código que permite inicializar la pantalla y posteriormente pasar a un estado de espera por datos provenientes de otro módulo (pudiera ser un teclado o un contador, como en el caso del diseño solicitado). Observa que cuando se inicializa la LCD, las variables *RELOJ* y *COUNT*, son sólo para el mensaje corto; una vez concluida la inicialización, ahora, las variables *DEBOUNCE* y *DIN* se convierten en el *enable* y el dato a escribir, respectivamente.

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
ENTITY ini_rel IS
  PORT(
         RELOJ, RESET, DEBOUNCE: in STD_LOGIC; --PRUEBA
         DIN: in STD_LOGIC_VECTOR (3 downto 0);
         PASSCLK: out STD_LOGIC;       -- Salida del PIN Enable de la LCD
         lcd: out STD_LOGIC_VECTOR (8 downto 0));
  END ini_rel;

ARCHITECTURE cont OF ini_rel IS
signal CLK, hab: STD_LOGIC;
signal COUNT: STD_LOGIC_VECTOR (3 downto 0);
signal ESCRIBE: STD_LOGIC_VECTOR (3 downto 0);
signal COUNTT: STD_LOGIC_VECTOR (4 downto 0);

begin
contador: process (CLK, RESET)
begin
   if RESET='1' then
      COUNT <= "0000";
       elsif CLK='1' and CLK'event then
         if COUNT /= "1010" then
               COUNT <= COUNT + 1;
         else
               COUNT <= COUNT;
         end if;
    end if;
end process contador;

mux: process (COUNT, RELOJ, CLK, DEBOUNCE, DIN)
begin
-- Este MULTIPLEXOR permite deshabilitar el RELOJ
if COUNT /= "1010" then
CLK <= RELOJ;
hab <= '0';
ESCRIBE <= COUNT;
else
CLK <= DEBOUNCE; -- En dado caso de que se requiera un reloj manual
hab <= '1';
ESCRIBE <= DIN;
end if;
PASSCLK <= CLK;
end process mux;


pantalla: process (COUNTT, COUNT, hab, ESCRIBE)
begin
COUNTT(4) <= hab;
COUNTT(3 downto 0) <= ESCRIBE;
--      RS|D7|D6|D5|D4|D3|D2|D1|D0
--      4 |14|13|12|11|10|9 |8 |7     "Pines de la LCD"
--      RS es el MSB de la palabra y se conecta al pin 4 de la LCD.
--      D7 es el MSB del dato y se conecta en el pin 14 de la LCD, por lo que el LSB
--      D0 se conecta al pin 7 de la LCD.
case COUNTT is
when "00000" => lcd <="000111000"; --038
when "00001" => lcd <="000000001"; --006
when "00010" => lcd <="000001110"; --00E
when "00011" => lcd <="000000100"; --004     Decrementa
when "00100" => lcd <="010000011"; --083     80 es la primera. Cuarta posicion
when "00101" => lcd <="100111010"; --148     ":"
when "00110" => lcd <="101000011"; --16F     "E"
when "00111" => lcd <="101000101"; --16C     "C"
when "01000" => lcd <="101010100"; --161     "T"
when "01001" => lcd <="100111010"; --13A     /* NO UTILIZADAS EN ESTE EJERCICIO*/
when "01010" => lcd <="101010100"; --154     *
when "01011" => lcd <="101000101"; --145     *
when "01100" => lcd <="101000011"; --143     *
when "01101" => lcd <="100100001"; --121     *
when "01110" => lcd <="100010000"; --110     *
```

```
when "01111" => lcd <="100010000"; --110      " "
when "10000" => lcd <="100110000"; --130      "0" Inicia conversion ASCII
when "10001" => lcd <="100110001"; --131      "1"
when "10010" => lcd <="100110010"; --132      "2"
when "10011" => lcd <="100110011"; --133      "3"
when "10100" => lcd <="100110100"; --134      "4"
when "10101" => lcd <="100110101"; --135      "5"
when "10110" => lcd <="100110110"; --136      "6"
when "10111" => lcd <="100110111"; --137      "7"
when "11000" => lcd <="100111000"; --138      "8"
when "11001" => lcd <="100111001"; --139      "9"
when "11010" => lcd <="101000001"; --141      "A"
when "11011" => lcd <="101000010"; --142      "B"
when "11100" => lcd <="101000011"; --143      "C"
when "11101" => lcd <="101000100"; --144      "D"
when "11110" => lcd <="101000101"; --145      "E"
when "11111" => lcd <="010000101"; --085      "Escribe posicion reloj en la 85"
when others => lcd <="100010000";  --110      " "
end case;
end process pantalla;
END cont;
```

Campo 4: **Conclusiones individuales.**